## ATM Networks → Stand for Asynchronous Transfer Mode

* ATM is a Connection oriented n/w it mean establish a Connection b/w Client & Server befor message or Data Transfer.

* ATM N/w work for cell relay that support voice, video and data Comm$^n$. It encode the data into small fixed size cell. & then transmit to physical medium.

* The size of ATM cell 53 Byte.

| Header<br>5 Byte | Payload<br>48 Byte |
|---|---|

* ATM used as Backbone of the public switched telephone network (PSTN), ISTN (Integrated Service Digital N/w.

* Reference model for ATM approximetly maps to three lowest layer of ISO-OSI.

| NL | DDL | PL |
|---|---|---|

* Provide functionality of Packet switching & ~~N~~ Circuit Switching.

* Encode Data into small fixed size packets.

Reference model data link layer (2), basic transfer unit called frame at D'LL

* In ATM these frame are of a fixed (53 Byte) Length & called cells.

* ATM Conn$^n$ oriented in which virtual circuit is established first b/w two end point.

Virtual Ciruit

- → Permanent ( Dedicated Conn$^n$ that are usually preconfigured by the Service provider
- → Switched. ( set up on a per-call basic using Signaling & disconn$^n$ when call is terminated.

* Two type of different cell formats :-

- → UNI (User Network Interface)
  - ↳ mostly used.
- → NNI (Network Network Interface)

UNI ATM cell

| 7 | | | 4 | 3 | | | 0 |
|---|---|---|---|---|---|---|---|
| GFC | | | | VPI | | | |
| VPI | | | | VCI | | | |
| VCI | | | | | | | |
| VCI | | | | PT | | CLP | |
| HEC | | | | | | | |
| Payload & Padding if. Necessary (48 Byte) | | | | | | | |

NNI

| 7 | | | 4 | 3 | | | 0 |
|---|---|---|---|---|---|---|---|
| VPI | | | | | | | |
| VPI | | | | VCI | | | |
| VCI | | | | | | | |
| VCI | | | | PT | | CLP | |
| HEC | | | | | | | |
| Payload & Padding if Nessary (48 Byte) | | | | | | | |

GFC = Generic flow Control

* 4 Bit field.
* Design to give the user Network interface 4 Bit in which to Negotiate multiplexing & flow control among cells of various ATM connections.
* Always ~~seto~~ set to 0000.

VPI → Virtual path Identifier.

↓ 8 Bit UNI

↓ 12 Bit NNI

VCI → Virtual Channel identifier (16 Bit)

PT → Payload type (3 Bit)

PT Bit3 (msbit) :- Network management cell.

PT Bit2 (Explicit forward CongestionIndication) } N/w Mgmt / Resource mgmt

EFCI1= N/w Cong. experienced.

PT Bit1 (lsbit) ATM user to user Bit.

CLP (Cell Loss priority) 1 Bit

HEC Header error Control (8 Bit)

PT field is used to designate various special kind of cells for operation adminstration & management purpose (OAM). to desitinate packet Boundaries. In AAL (ATM adaption Layer.)

UNI→ Reserve the GFC field for a local flow control / sub multiplexing System b/w user.

* This is allow several terminal to share single N/w Connection. like ISDN share single basic rate Connⁿ.

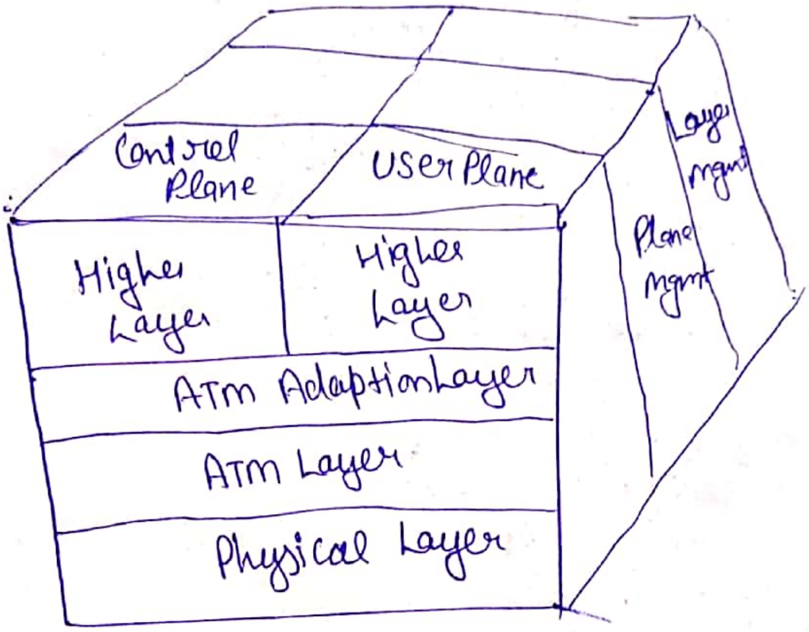
* Default 4 Bit set to 0000.

NNI→ , cell format replicates UNI format almost , except 4 Bit GFC field is reallocated to VPI field,

~~* Extend Bit GFC~~

Extend VPI to 12 Bit.

**ATM Reference Model→**



**Physical Layer→** manage medium - dependent Transmission.

**ATM Layer→** Together with ATM Adaption Layer

→ maps to DLL of OSI model

→ Responsible for Connection establishment, cell multiplexing cell Relay.

Responsible for
- Connection setup
- multiplexing Demultiple. of cell.

**AAL→** togethe with ATM Layer Adapts (Segement) user data from Higher Layer protocol into 48 Byte cell payloads.

**Higher Layer→** Accept user data form it into packets and pass packet Down to AAL.

I. Middleware Distributed System:- Is a S/w that provide services beyond those provided by OS. to enable the various Components of distributed System to Communicate and manage data.

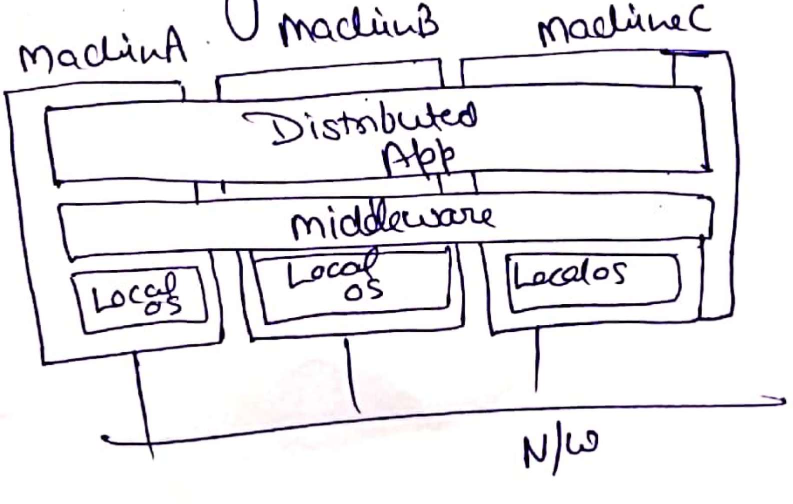
* It is a S/w layer that lies b/w the operating system and the application on each side of a distributed Computer Network.

* In distributed System it Hide the distributed nature of Application.

* Support and Simplifies Complex distributed App

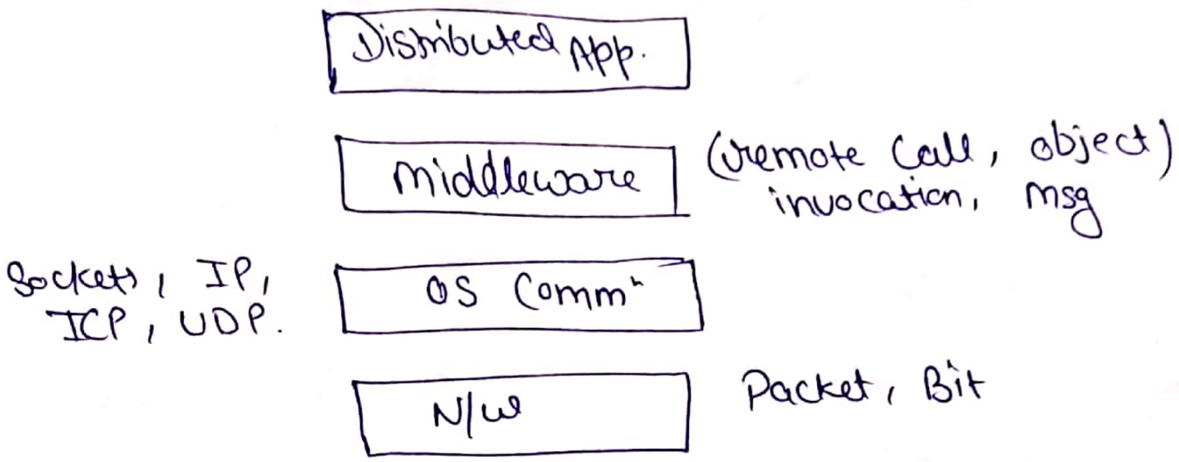
* It include web server, application server, messaging and similar tools that support application development and delievery.

*



Hide complexity and Heterogenity of system.

- Bridges a gap between low level OS Communication and programming language and abstraction.



RPC→ Hide communication detail behind a procedure call.

Sockets→ OS level interface to underlying commn protocols.

TCP/UDP, IP } → underlying commn protocols.

Uses of middleware→

- Locate transparently across the n/w thus providing interaction with another service or App.
- Filter Data to make them friendly.
- Independent from N/w Services.

## Integeration Levels :-

- Data Integration → Integration data resours like file and DB.
- Cloud Integration → Integration b/w various Cloud Services
- B2B → Between Data resourse and parters interface.

**Type of middle ware →**

- → message Oriented
- → Intelligent
- → Content - centric Middleware

**Message oriented Middleware** → where transactions or event notification are delivered b/w disparate System or Components by way of messages.

* By MOM msg Sent to Client are collected and Stored until they are acted upon , while the Client continue with other processing.

ESB (Enterprise Service Bus).

support Both

MOM

web sewice

Intelligent Middlewares (IMW)

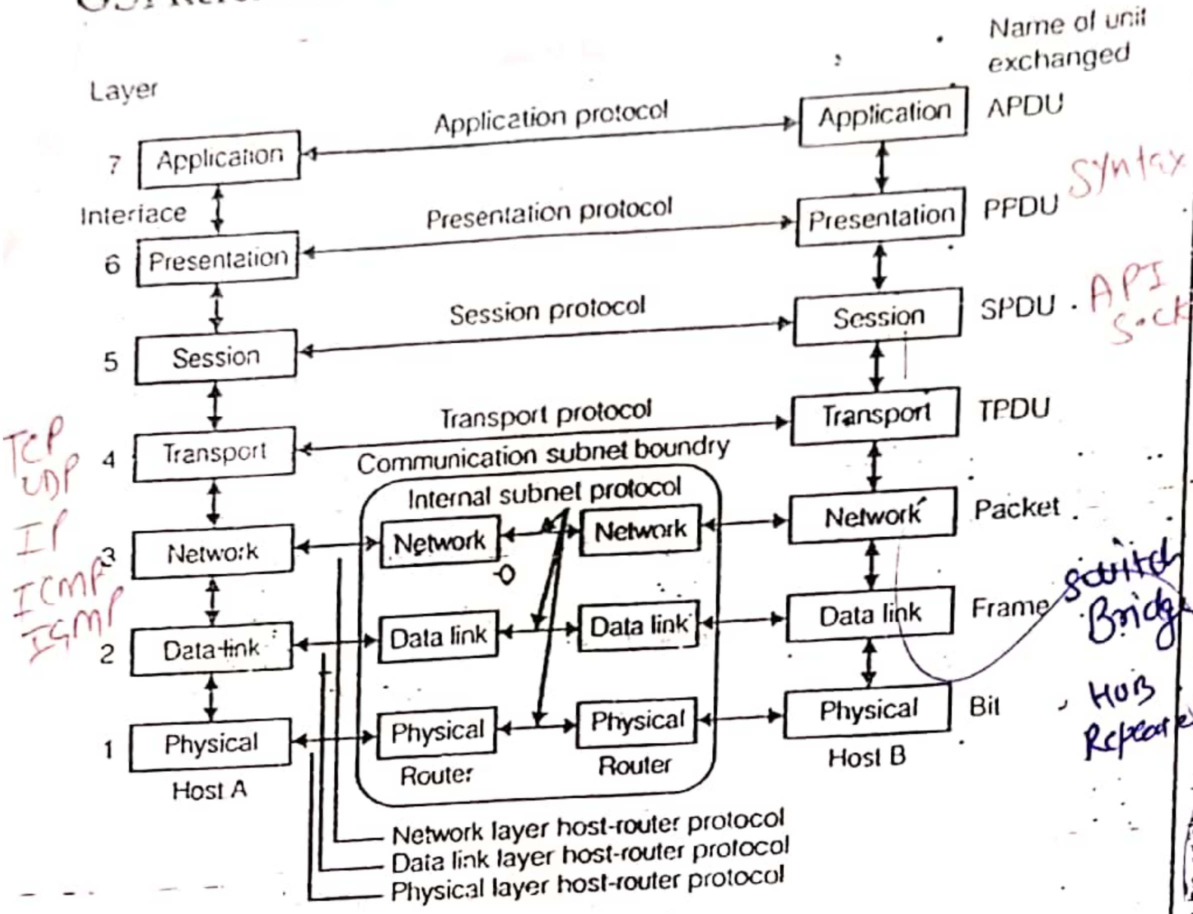
provide real time intelligence and event management through intelligent agents.

* Manage real time processing of High Volume sensor signals and turns these signals into intelligent and actionable business information.

## 3 Content - Centric Middleware

It offer simple provider- consumer abstraction through which application can issue request for uniquely content, without worrying about where or How it is obtained.

# OSI Reference Model

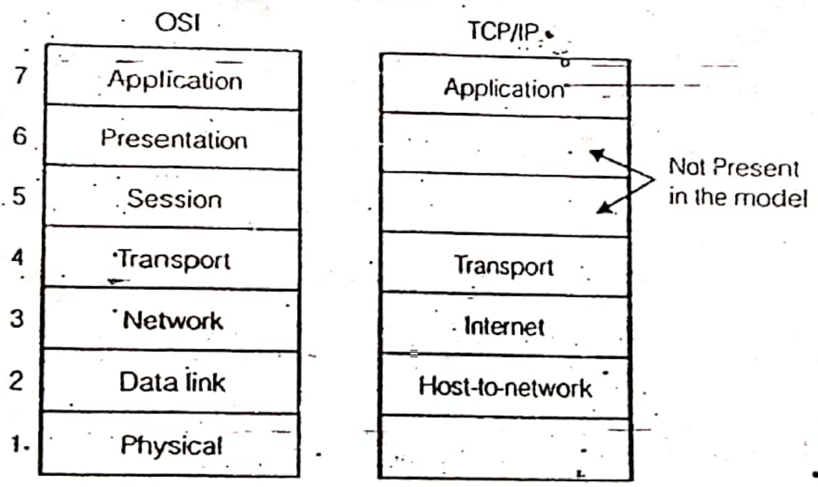


The OSI reference model

- Physical Layer:
  - Data conversion in the form of electric pulse
  - Setting an agreement between the receiver and NIC.
  - Design issues deal with mechanical, electrical and procedural interface and physical transmission medium.
- Data Link Layer:
  - Framing
  - Error control
  - Flow control -fast sender and slow receiver or vice-versa
  - In case of noise burst the frame has to be retransmitted.
- Network Layer:
  - Transforming logical address to physical address.
  - Routing data (how much data to carry and how much network traffic).
  - How packets are routed from source to destination.
  - Controlling congestion (If too many packets are present in the network at the same time)
  - Allow heterogeneous networks to be inter connected.
- Transport Layer:
  - Sequencing, desequencing and segmenting.
  - Accept data from session layer, split if required, send to the network layer and ensure pieces all arrive correctly at the other end.

- Multiplexing several transport connections on same network connection.
- It is a true end-to-end layer from source to destination. In other words, a program on the source machine carries on a conversation with a similar program on the destination machine.
- Takes care of establishing and deleting connections across the network.
- Flow control between hosts.

- Session Layer:
  - Establishing link or sessions
  - Synchronization
  - Token Management
- Presentation Layer:
  - Maintains security, encryption, decryption
  - Concerned with syntax and semantics of the information transferred.
- Application Layer:
  - Interface between user and OSI
  - Network virtual terminal
  - File transfer



## TCP/IP Reference Model

- Internet Layer:
  - Packet switching network based on a connection less internetwork layer. This layer is called Internet layer.
  - Its Job is to permit hosts to inject packets into any network and have them travel independently to the destination (potentially an a different network.
  - The Internet layer defined an official packet format and protocol called IP (Internet Protocol). The job of the Internet layer is to deliver IP packets where they are supposed to go.
  - Packet routing and voiding congestion are major issues.
- Transport Layer:
  - Designed to allow peer entities on the source and destination hosts to carry on a conversation.

- Two Protocols have been defined here –
  - TCP (Transmission control Protocol): The transmission control protocol is a reliable connection-oriented protocol that allows a byte stream originating on one machine to be delivered without error on any other machine on Internet. It fragments the incoming byte stream onto discrete messages and passes each one on to the Internet layer. At the destination, the receiving TCP process reassembles the received messages into the output stream. TCP also handles flow control to make sure a fast sender cannot swamp a slow receiver.
  - UDP (User Datagram Protocol): The second protocol in this layer, UDP(User Datagram ~~Protocol~~), is an unreliable, connectionless protocol for applications that do not want TCP's sequencing or flow control and wish to provide their own. It is also widely used for one-shot, client-server type request-reply queries and applications in which prompt delivery is more important than accurate delivery such as transmitting speech or video.
- Application Layer:
  - On the top of transport. Layer is Application layer.
  - contains all the higher level protocols.
  - TELNET, FTP, SMTP, DNS, TFTP, ICMP, etc protocols are used.
  - ICMP: When something unexpected occurs it is reported by ICMP which is used to test the Internet.

## Similarity between OSI and TCP/IP

- Both are based on concept of a stack of independent protocols.
- Functionality of layers is roughly similar.

## Difference

- OSI: Network layer supports both connection oriented and connectionless, transport only connection oriented.
- TCP/IP:
  - Network: only connection less.
  - Transport: connection oriented and connection less and gives a choice to user.

OOOO

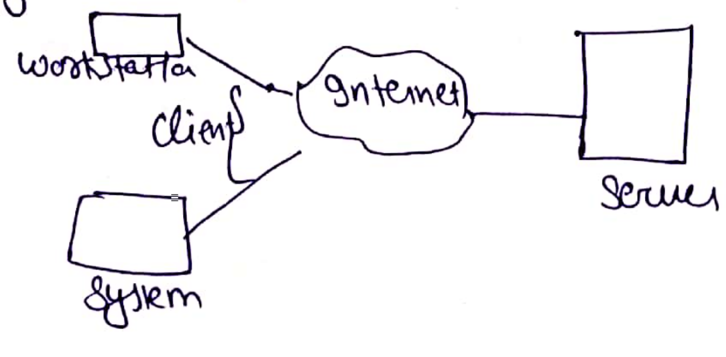
**Client | Server Model** → The client model is a distributed application structure that paritions task or workload b/w the povider of resourse or service. Called server, service requester called client.

* In client server architecture, when a client computer send a request for a data to server through the Internet.

* The server accepts the requested process and deliver data packets requested back to client.

* Ex:- Email, www etc.

* Client server connection establish through a n/w.



- A Server Host run one or more server program which share their resources with client.

* Client - Server model are core n/w computing concept also building functionlity for email - exchange and web / database access.

* Poretocols Built around client / server Model

  - → Hypertext Transfer Poretocol.
  - → Domain Name System
  - → SMTP
  - → Telnet.

Client include
- → web browser
- → Chat applications
- → Email S/w

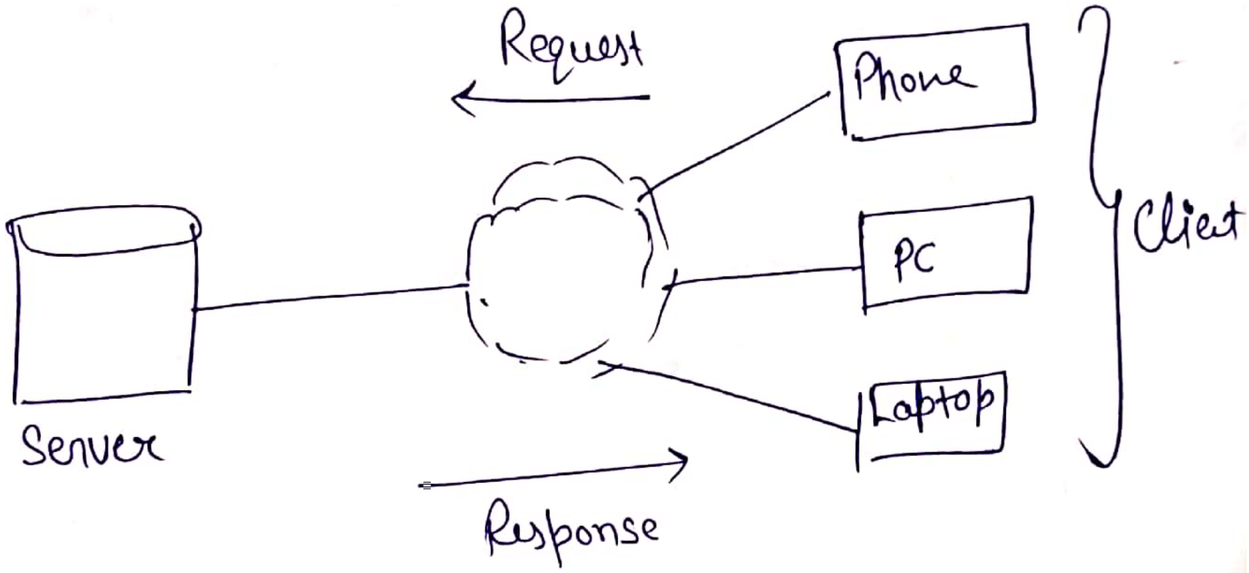
Server Include
- → web
- → Database
- ~~→ chat~~

⇒ How Client / Server model work:-

Client → The word client mean to talk of person or an organisation using particular service.

In digital world a client is computer Host i.e. capable of recieving information or using particular service from service provider.

* **Server** → It mean person or medium that serve Something. Server is also remote computer which provide information (Data) or access to particular service.



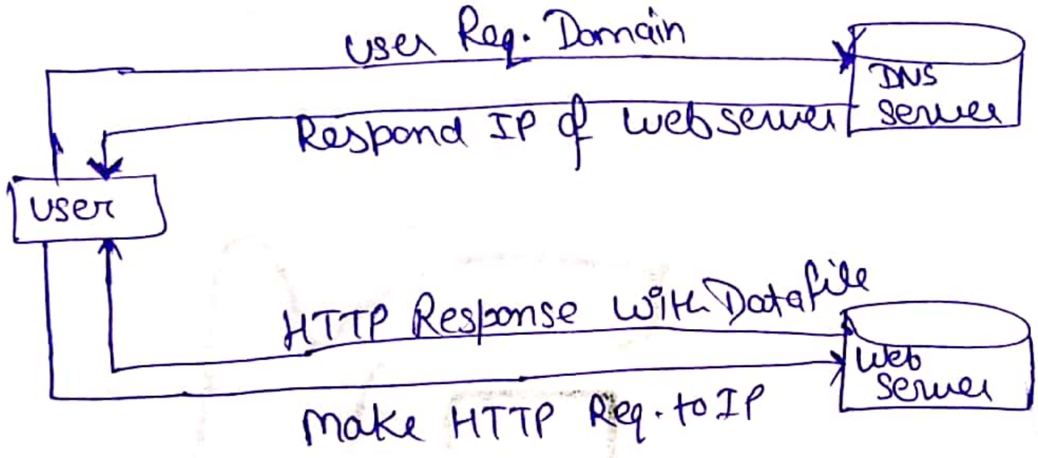
**How Browser interact with server** →

* User enter the URL (uniform Resource Locator) of the website or file.
* The Browser than request the DNS.
* DNS Server looks up for the address of web server.
* DNS respond with IP of web server.

Browser Send over an HTTP request to web server IP

* Server Send over the necessary file of the website.

* Browser render file and the website display.



Scanned by CamScanner

* Clock Synchronization → For timer mechanism use Computer clock to keep track of current time and also various accounting purpose such as:-

- Calculating the time spent by a process in CPU ultilization,
- Disk I/o and corrosponding user can be charged properly.
- In distributed System an application may have processes that Concurrantly run on multiple node of the System. Distributed app. require Clocks of the nodes are synchronised with each other.
- Difficult to get the correct result in the case if the Clock of the sender and reciever node are not synchronised.
- Distributed System may have no physical Synchronous global Clock, So a logical clock allow global ordering on event from different processes in such Systems.

→**How Computer Clock are Implemented:-** A Compu Clock consist of 3 Components →

- A quartz that oscillates at well define frequency.
- A Counter Register
- Constant Register. → use to store constant value that is decided and based on frequency of oscillations of quartz crystal.

**Counter Register** :- Value of CR is decrement by 1 ~~after~~ for each oscillation of quartz crystal. when CR value become 0 an interrupt is generated and its value is reintialized to the value of constant Register.

- { Each interrupt is called clock tick. }

Date:

**Drifting of clock** → A Clock always run at a constant rate because it quartz crystal oscillates at well define frequency.

Due to differences in crystals, the rate at which two clock run are normally different from each other.

* Therefore a Computer Clock drift from real time clock that was used for its intial Setting

* clock drift mean where a clock does not run at exactly the Same rate as reference clock. After Some time clock " drift apart" or gradually desynchonize with other clock.

* Suppose when real time is $t$, time value of clock $p$ is $C_p(t)$. All the clock were perfectly Syncheized and all $t$. we have $C_p(t) = t$ for all $p$

- If $C$ denote time value of a clock ideal case $dC/dt$ Should be 1.

if max. drift rate allow is $\rho$ S

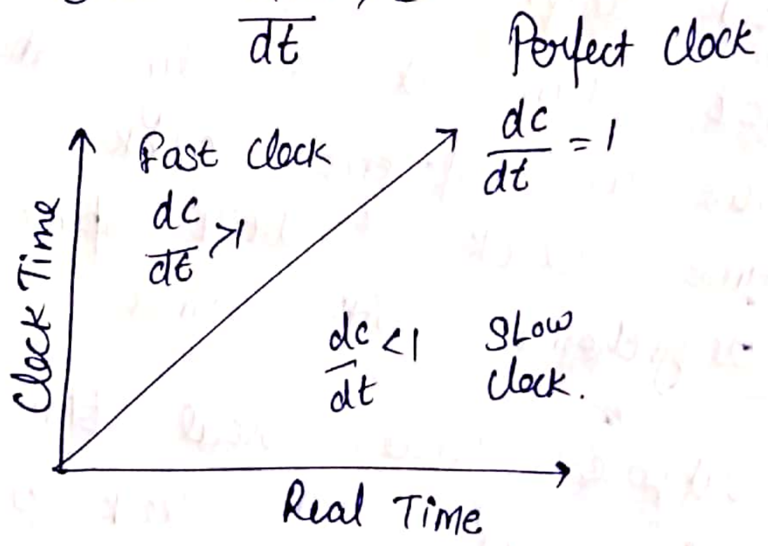
a ~~Celock~~ Clock Said to be nonfaulty if following Condition Hold.

$$1 - \rho \le \frac{dc}{dt} \le 1 + \rho$$

* After Synchanization with perfect Clock Slow and fast clock drift in opposi directions from the perfect Clock.

Slow Clocks $\frac{dc}{dt} < 1$

Fast Clock $\frac{dc}{dt} > 1$



* Node of DS are periodically resynchaize their Clock to maintain a global time base across entire System.

* Fast clock drift opposite direction of perfect clock.

Synchronization of Computer Clock with real-time (external Clock):-

- Require for real-.time applications.
- External Clock Syn. allows the System to exchange information about the timing of event with each other System & user.

2 Mutual (Internal) syn. of the Clock of different node of the System:- use for app. require consistent view of time across all node of distributed System

* Converse is not true bcz with the passage of time internally Synchronised clock may drift arbitrarily far from external time.

⇒ Clock Synchronization issues:-

* Two clock are said to be Syn. at at a particular instant of time if the difference in time values of two less than some specified Constant δ.

* The difference in time two clock value are called <u>Clock Skew.</u>

* Clock syn. need or require each nod to read the other node Clock val Error occure because unpredictable commn delays during message passing used to deliver clock Signal or Clock msg from one node to another node.

* Error occur , msg transmitted several time , random event occure such as page fault , process switch.

* Time must Never run backward , such a repetition of certain operations that may be costly.

* <u>Clock</u> <u>Syn.</u> <u>Algorithium</u> →
  - → centeralized algo
    - Passive Time Server
    - Active Time Serve
  - → Distributed algo.
    - global averaging
    - Local Averaging.

Clock

Centralised algo" → One Node has a real time reciever. This Node called time - Server Node. Clock time of this node is regarded as correct and used as reference time.

* goal of this algo is keep clock of all other node synchronized with the clock time of time server Node.

* Passive Time Server → Each Node periodically send a massage ("time = ?") to the time server.

- when time server recieve a msg its quickly response with msg ("Time = T")

↑ current time in clock of Time Server Node.

* when a client Node send the time "time = ?". msg its clock time is To and when it recieve the "time = T" msg it clock time is $T_1$.

* So To, $T_1$ are measure using same clock.

* From the time Server Node to the Client Node is $(T_1 - T_0)/2$ Therefore when reply recieved at Client Node, its Clock is readjusted to $T + (T_1 - T_0)/2$.

⇒ **Active Server Algo**ⁿ → In passive time server approach the time Server only respond to request for time from other Nodes. In active approach time server periodically broadcast its Clock time ("time = T").

* Other Node recieve broadcast msg and use the Clock time in the msg for correcting their own Clock.

* Each Node has a priori Knowledge of the appromate time (Ta) required for the propagation of msg "Time = T" from Server Node to its own Node.

* when Broadcast msg is recived at a Node, the Node clock is re-adjusted to time T + Ta.

* Not Fault Tolerant Drawback.

* Resyn method is broadcast from each Node at the begin of every- fixed Length resynchronization interval.

* After Broadcasting the clock value, the clock process of a node wait for time T, where T is parameter to be determine by the algo^n.

* During this waiting period, clock process collect the resyn msg broadcast by other Node.

* For each resync msg the clock process record the time, according to its own clock, when msg recieved.

⇒ Localized Averging :- Global require the N/w to support broad cast facility and also large amount msg traffic generated.

* Node of distributed System are logically arrange in some Kind of pattern, Such as ring or grid.

**Drawback Centralized →**

1. Subject to Single point failure :- If the time Server Node fail clock Syn operatio Not performed.

2. Scalibility

{ Not acceptible to get all the time req. service by a Single Time Server. }

⇒ **Distributed Algo:** A Simple method of clock Synⁿ may be equip each node of the system with a real-time reciever so that each Node clock can be independently Synⁿ with Real time.

* Multiple real-time clock (one for each Node) used.

⇒ **Global Averaging DA:** The clock process at each Node broadcast its local clock time in the form of special "resync" msg when its local time equal to

$T_0 + iR$ → system parameter (depen upon No. of Node in sys.

↑ Fixed time

↓ Some Integer

Periodically each Node exchange its Clock time with its Neighbours in ring, then set its clock time to average of its own Clock time and Clock time of its Neighbours.

**Atomic Transactions** → Atomic Database transaction in which indivisible and irreducible series of database operations such that either all occur, or Nothing occur.

* They require the programmer to be intimately involved with all detail of mutual exclusion, Critical section, deadlock prevention and crash recovery.

* Higher Level Abstraction, one that Hides these technical issues and allow programmer to concentrate on the algorithium and How the work together in parellel.

* Transactions Help to preserve the Consistency of set of Shared object in the face of failure and the concurrent access.

Transaction end in two State

→ Transaction carried out Completely

→ Transaction fail Completely.

## * Transaction Properties →

A C I D property

- A → Atomocity
- C → Consistancy
- I → Isolation
- D → Duraibility.

1. Atomocity → (Failure of atomocity) :- It ensure that to the outside world all operation of a transaction appear to have been performed indivisibly.

Two Req:-

- → Atomocity with respect to failure
- → Atomocity with respect to concurrent acces.

A Failure atomocity is also known as all - or Nothing property.

2. Concurrancy atomicity ensure that while a transaction is in ~~progess~~ progress, other process executing concurrently with the transaction cannot modify or observe intermediate state of transaction.

only final state become visible to other processes after transaction Complete.

~~3 Permanence,~~

3 **Isolation (serializability) :-** Ensure that concurrently executing transaction donot interfare with each other.

* Two or more than two transaction are councurrently executing in serial equivalent.

4 **Durability (Permanence) →** Ensure that once a transaction Complete Sucessfully, the result of its operation become permanent and cannot lost even if the corrospending process or the processor on which it run crashes.

Periodically each node exchange its clock time with its neighbours in ring, then set its clock time to average of its own clock time and clock time of its Neighbours.

⇒ <u>Mutual Exclusion</u> → There are several resources in a system that must not be used simultaneously by multiple process if program operation is to be correct.

<u>For Example</u>:-

* File must Not be simultaneously updated by multiple processes.

* Use of unit record peripheral such as tape drives or printed must be resticted to a single process at a time.

* Therefore exclusive access to such a shared resource by a process must be ensured.

* Exclusiveness of access called mutual exclusion b/w processes.

* The Section of a program that Need exclusive access to Shared resou are referred to critical Section.

* Mutual exclusion prevent process from executing concurrently within their associated critical Section.

* Requirement :-

→ Mutual Exclusion

→ No Starvation.

Mutual Exclusion: Given a Shared resource accessed by multiple Concurrent process. at any time only one process Should access the resourse. That is the process that has been granted the resourse must release it before it can be granted to another process.

No Starvation: If every process that is granted resourse eventually release it, every req. must be eventually granted.

3 Basic Approach

- → Centoralised
- → Distributed
- → Token passing

⇒ Centoralised Approach →

* One of the processes is system elected as coordinator and coordinates the entry to critical section.

* Each processwantenter to enter a critical section must first seek permission from the coordinator.

* If No other process is currently in critical section must coordinator can immediately grant permission to Req. process.

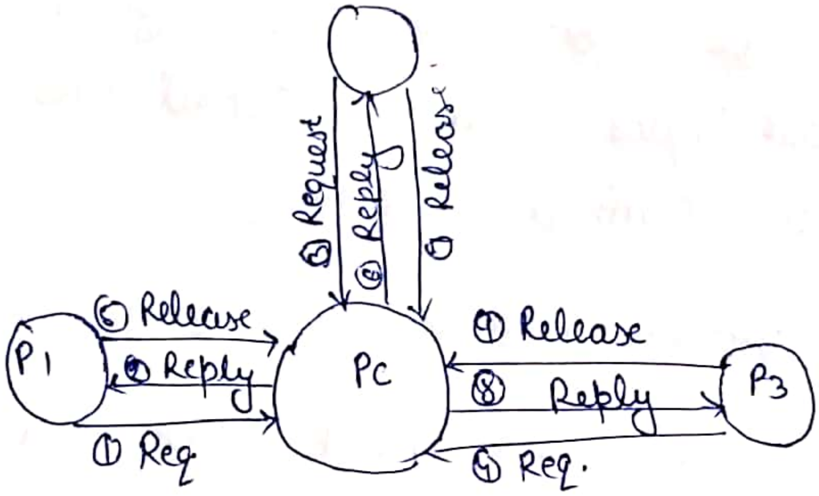
* If two or more processes concurrantly ask for permission to enter the same critical section., the coordinator grants permission to only one process at a time in accordance with some scheduling Algo.

* After executing critical section,

when a process exit critical section, It notify the Coordinator, So Coordinator grand permission to requested process

<u>Example:-</u>

*



Pc → Process Coordinator

P1, P2, P3 are processes in system.

* Request are granted in FCFS for which Coordinator maintain queue.

* P1 wants to enter critical section for which It sends request to Pc.

* On recieving P1 req. Pc check to see whether some other process is currently in CS.

* If No the Pc immedialtly send reply to P1. after reply P1 enter CS.

Suppose $P_1$ in critical Section $P_2$ wants to send request for permission to enter CS.

* $P_1$ is already in the critical section. $P_2$ cannot be granted permission.

* $P_c$ does Not Send a reply to $P_2$ Immediately enter its req. in Req. queue.

* Suppose $P_3$ is Still in the critical section
$P_3$ also sends a request massage to $P_c$ asking for permission to enter same critical section. No Reply send to $P_3$ by $P_c$. request is in queue.

* suppos $P_1$ Send a exit to CS and send release massage to $P_c$. On recieveing msg release, $P_c$ take the first req. from the queue of deffered & send reply msg to requested process.

* This algo is mutual exclusion bcause at a time, the coordinator allow only one process to enter critical section.

**Distributed Approach** → All the processes that want to enter the same critical section cooperate with each other before reaching a decision on which process will enter CS Next.

* When a process want to enter a CS, it send a request message to all other process. The msg contain:-

① The process identifier of process.

② The name of the critical section that process want to enter.

③ Unique timestamp generated by the process for Request massage.

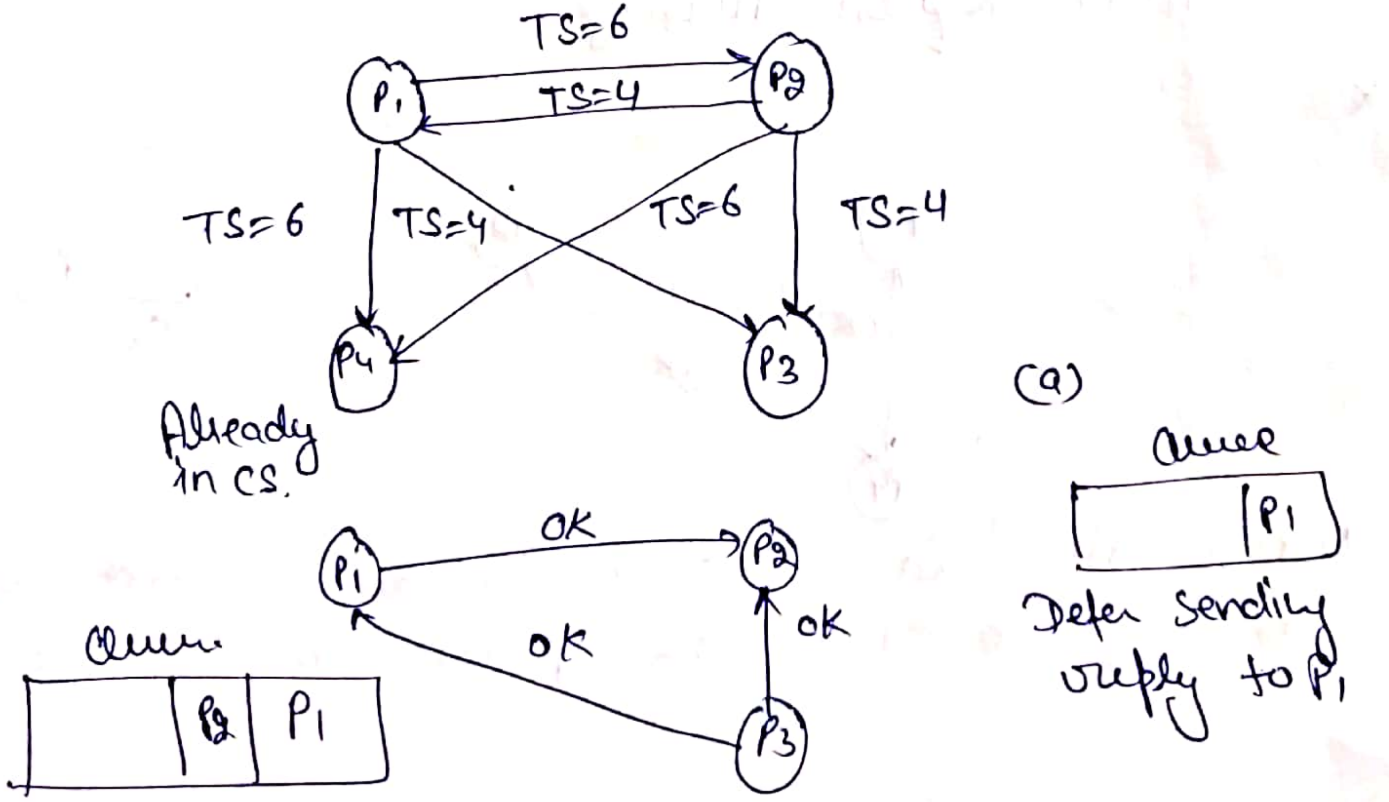
→ On Recieving Request Massage, a process either immediately send back a reply msg to sender or defers sending reply base on rules:-

I If the reciever process itself currently executing in CS. it simply queue the request msg and defers sending a reply.

It the reciver process Not con currently executing in critical section but is waiting for its turn to enter CS. it Compare the time stamp in its Req. recieved msg. with the time stamp of its own request massage.

If timestamp of recieved request massage is lower, it mean sender process made request before reciever process to enter CS.

3 If reciever process neither is in critical or Nor in waiting it immediately send back reply msg.



Example $P_1$, $P_2$, $P_3$, $P_4$ processes.

* $P_4$ → in critical Section

* $P_1$, $P_2$ → want enter to critical section.

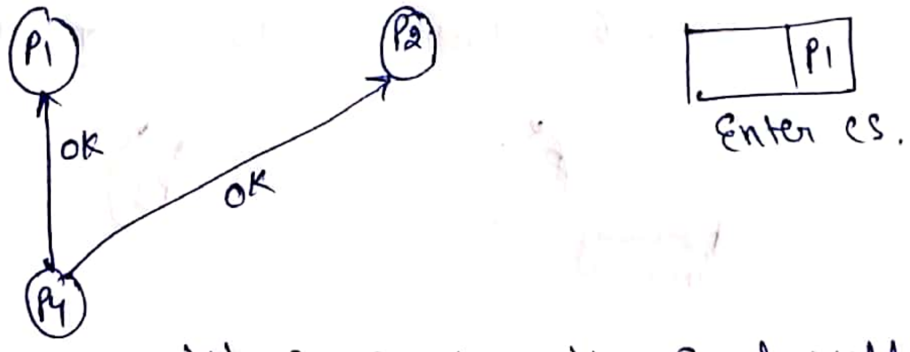
* Get permission $P_1$, $P_2$ send request msg with timestamp· TS 6, 4 to other process.

* $P_4$ is already in CS it defer send msg to $P_1$, $P_2$ enter them in queue.

* $P_3$ is Not interest to enter CS So it send ok msg to $P_1$, $P_2$. (b)

* $P_2$ defer send a reply msg to $P_1$ and enter $P_1$ queue because timestamp (4) in its req. massage less timestamp(6) in $P_1$ Request massage.

*



When $P_4$ exit critical Section it send reply massage to all the process in its queue.

P2 reciene massage from (P1, P3, P4)

P2 enter to critical section.

* when P2 exit CS send reply msg to P1.

* Algorithium require that each process know the identity of all process participating in mutual-exclusion algo.

* Process enter to CS after dealing with all other processes and get permission from them.

3 **Token Passing** → mutual exclusion achieved by using single token that is circulated among the processes in the system.

* A token special type msg that entitles its Holder to enter critical section.

* Processes are organised in Ring structure. token is circulated from one process to another. (clock wise & Anticlockwise).

* If process Not want to enter critical section pass token ~~to~~ ring to its Neighbour Process.